دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

کارشناسی ارشد علوم کامپیوتر گرایش داده کاوی

پروژه شماره ۶ درس داده کاوی

نگارش

حدیث حق‌شناس جزی

استاد راهنما

دکتر مهدی قطعی

استاد مشاور

آقای بهنام یوسفی مهر

دی ۱۴۰۱

چکیده

در این گزارش هدف این است که با استفاده از دیتای CC GENERAL که شامل داده هایی از کارت های اعتباری ۹۰۰۰ شخص است که در ۶ ماه جمع آوری شده است، ابتدا به پیش پردازش داده بپردازیم و سپس به کمک ۳ نوع خوشه بندی انتخابی K-means و Agglomerative Clustering و DBscan ابتدا به بررسی و انتخاب تعداد خوشه بندی و تنظیم هایپر پارامترها پرداخته و سپس بر روی بهترین نتایج حاصله، خوشه بندی داده را به بر روی نمودار نشان داده و نهایتا این ۳ خوشه بندی را با ۳ معیار مقایسه، مقایسه کنیم.

فهرست مطالب

فصل اول

مقدمه

مقدمه

به منظور مقایسه روش های خوشه بندی ذکر شده به معرفی ۳ نوع شاخص با نام های Davies-Bouldin و Silhouette Score و Calinski-Harabasz میپردازیم و سپس با شناخت بر این موضوع که در هر خوشه بندی چندین پارامتر تعریف میشود و در میان آنها بعضی پارامتر ها از اهمیت بیشتری برخوردار هستند، به مقایسه و انتخاب بهترین اندازه هایپر پارامتر های مختص هر خوشه بندی پرداخته و سپس نتایج حاصله را مقایسه میکنیم.

فصل دوم

پیش پردازش داده ها

پیش پردازش دیتاست

در ابتدا فایل دیتا را فراخوانی میکنیم. داده شامل ۸۹۵۰ سطر در ۱۸ ستون میباشد. اطلاعات کلی دیتاست در تصویر زیر قابل مشاهده است:

```
import pandas as pd
import numpy as np
import sklearn
```

```
df = pd.read_csv("CC GENERAL.csv")
```

```
df
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQ
0	C10001	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.
1	C10002	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.
2	C10003	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	0.
4	C10005	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.
...	...	...	...	...	...	...	...	
8945	C19186	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.
8946	C19187	19.183215	1.000000	300.00	0.00	300.00	0.000000	1.
8947	C19188	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.
8948	C19189	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.
8949	C19190	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	0.

8950 rows × 18 columns

حالا با دستور describe اطلاعات آماری و چارک داده هارا مشاهده میکنیم که در خصوص ستون "MINIMUM_PAYEMENTS" در ادامه به آن نیاز پیدا خواهیم کرد. همچنین با توجه به میانگین های متفاوت داده ها، پر واضح است که داده های مقیاس های عددی متفاوتی دارند (از بین ۰ و ۱ تا میانگین ۸ هزار و ...) به

```
df.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112	0.490351
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000	0.083333
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000

```
1 df.describe()
```

CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
8950.000000	8950.000000	8950.000000	8949.000000	8950.000000	8637.000000	8950.000000	8950.000000
0.135144	3.248827	14.709832	4494.449450	1733.143852	864.206542	0.153715	11.517318
0.200121	6.824647	24.857649	3638.815725	2895.063757	2372.446607	0.292499	1.338331
0.000000	0.000000	0.000000	50.000000	0.000000	0.019163	0.000000	6.000000
0.000000	0.000000	1.000000	1600.000000	383.276166	169.123707	0.000000	12.000000
0.000000	0.000000	7.000000	3000.000000	856.901546	312.343947	0.000000	12.000000
0.222222	4.000000	17.000000	6500.000000	1901.134317	825.485459	0.142857	12.000000
1.500000	123.000000	358.000000	30000.000000	50721.483360	76406.207520	1.000000	12.000000

همین دلیل از این اطلاعات متوجه میشویم که باید داده هارا حتما برای مراحل بعدی نرمال سازی کنیم تا بتوانیم مدل های خوشه بندی و همچنین الگوریتم کاهش ابعاد را به نحو بهتری نتیجه ببینیم:

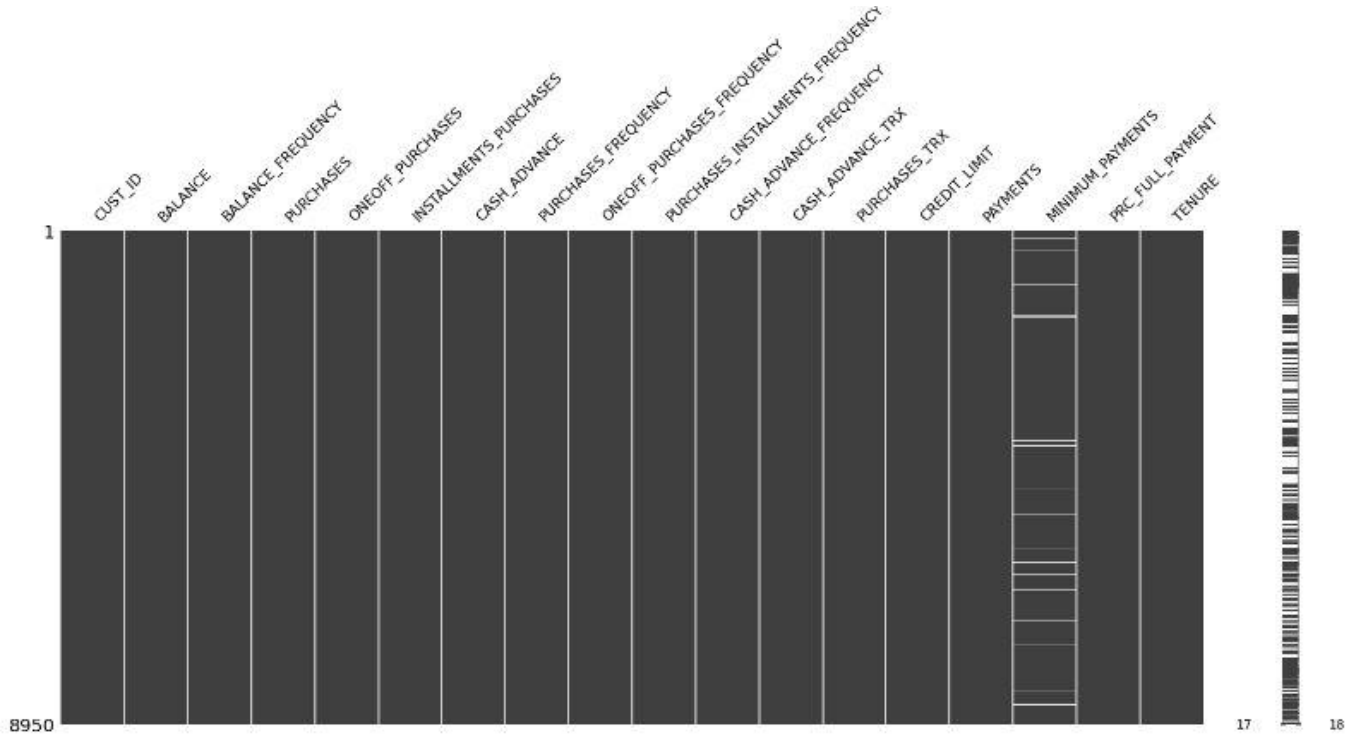
در ادامه به بررسی نوع داده ها میپردازیم. ستون اول ما که داده های کیفی هستند نشان دهنده آیدی کاربران کارت های اعتباری میباشند و مورد استفاده ما نیستند و در ادامه نیز برای الگوریتم ها به مشکل میخوریم پس ستون اول داده هارا در ادامه حذف میکنیم.

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

پر کردن داده های خالی

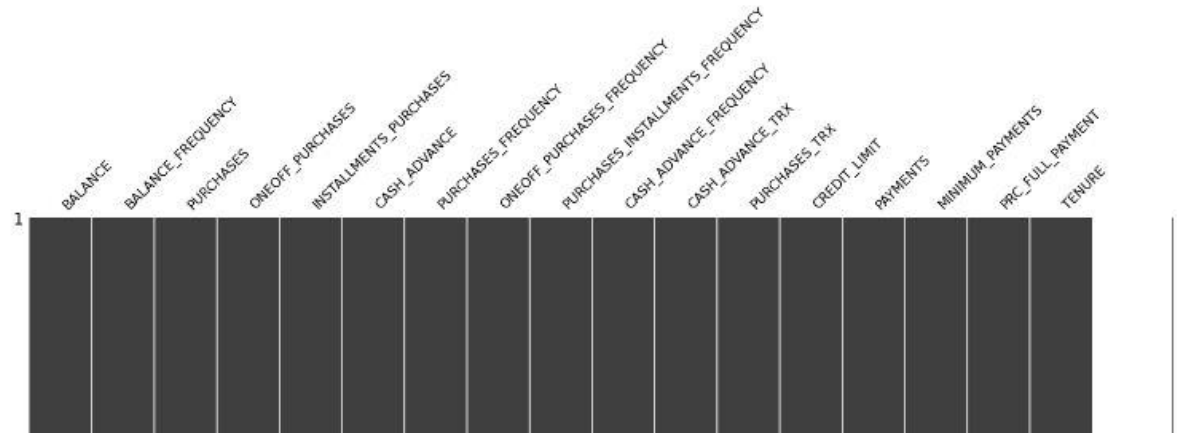
سپس به بررسی داده های خالی یا missing data میپردازیم. همانطور که مشاهده میشود مقادیر داده های خالی ما زیاد نیست و ستون های زیادی را درگیر نکرده است پس برای رفع این مشکل بهتر است به جای حذف سطر های خالی، به پر کردن سطر های خالی به کمک Imputer بپردازیم.



```python
df_copy = df
df= df.drop(columns= ["CUST_ID"])
```

```python
from sklearn.impute import SimpleImputer

# چون میانگین و چارک دوم خیلی از هم فاصله دارن مد رو میزاریم نه میانگین #
imputer= SimpleImputer(strategy='median')
df = imputer.fit_transform(df)
```

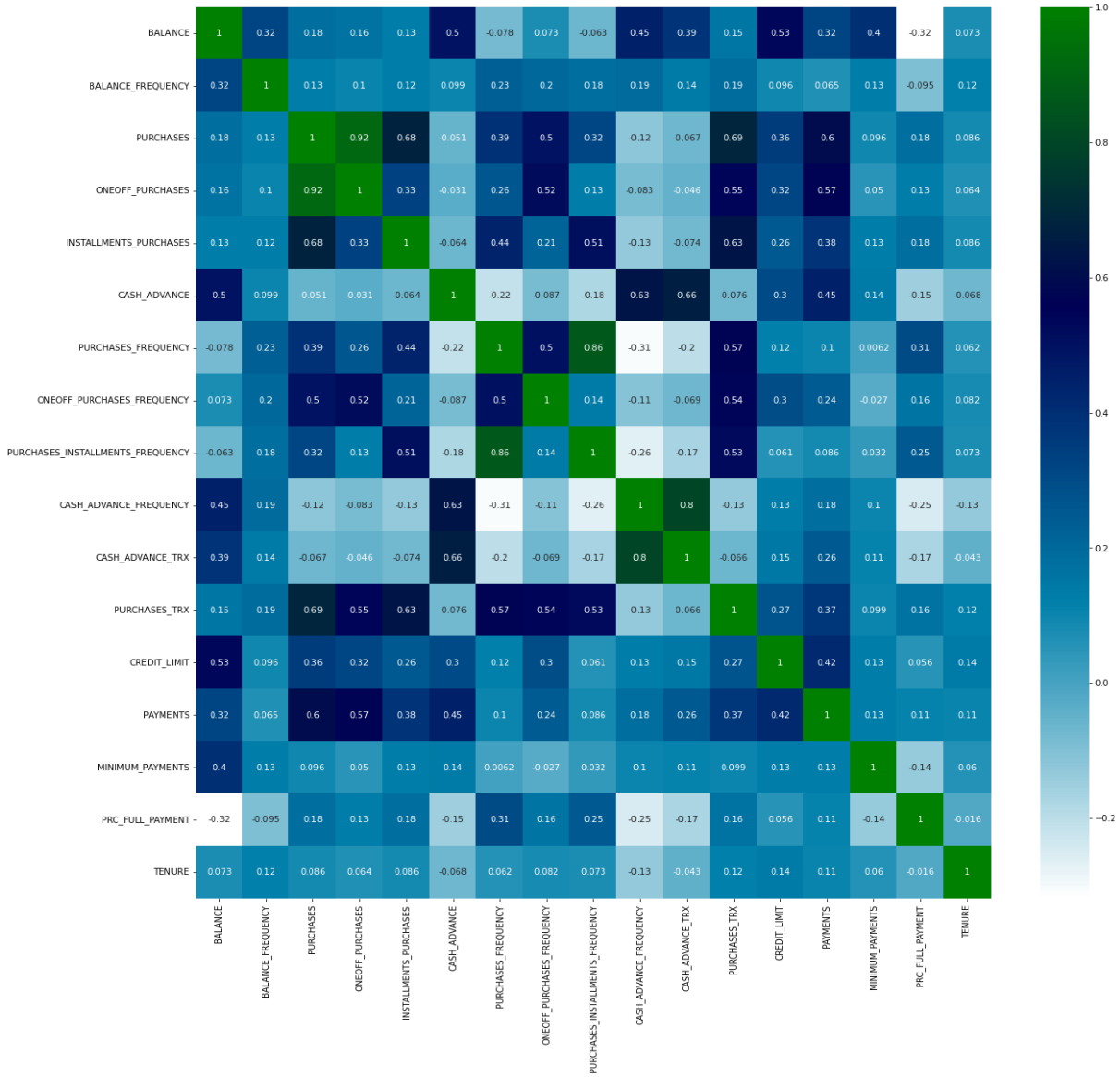
```python
df = pd.DataFrame(df, columns=['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
       'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
       'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
       'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE'])

msno.matrix(df)
plt.figure()
plt.show()
```



همانطور که در تصویر بالا مشخص میباشد به کمک Imputer و با مقادیر median این داده های خالی را در ستون MINIMUM_PAYMENTS پر میکنیم زیرا در این داده با توجه به مقدار میانگین و مد بهتر است که با مد پر شود تا داده های خیلی پرتی تولید نشود.

بررسی همبستگی ویژگی ها

حالا به کمک ماتریس همبستگی نمودار زیر را رسم میکنیم. در اینجا ۳ جفت ستون وجود دارد که میزان همبستگی آنها بزرگتر مساوی ۰,۸ است لذا از هر جفت ستون ها، یک ستون را یعنی در مجموع ۳ ستون را حذف میکنیم زیرا این ستون ها اطلاعات بسیار مشابه و در جهت آن ۳ ستون دیگر میباشد و نگهداری آنها توصیه نمیشود. سپس در تصویر بعدی اطلاعات کلی دیتا نشان داده شده است که ۱۴ ستون باقی مانده است.



```
df= df.drop(columns=["ONEOFF_PURCHASES","PURCHASES_INSTALLMENTS_FREQUENCY","CASH_ADVANCE_TRX"])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 14 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   BALANCE                           8950 non-null   float64
 1   BALANCE_FREQUENCY                 8950 non-null   float64
 2   PURCHASES                         8950 non-null   float64
 3   INSTALLMENTS_PURCHASES            8950 non-null   float64
 4   CASH_ADVANCE                      8950 non-null   float64
 5   PURCHASES_FREQUENCY               8950 non-null   float64
 6   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 7   CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 8   PURCHASES_TRX                     8950 non-null   float64
 9   CREDIT_LIMIT                      8950 non-null   float64
 10  PAYMENTS                          8950 non-null   float64
 11  MINIMUM_PAYMENTS                  8950 non-null   float64
 12  PRC_FULL_PAYMENT                  8950 non-null   float64
 13  TENURE                            8950 non-null   float64
dtypes: float64(14)
memory usage: 979.0 KB
```

حالا مجددا به دلیل وجود داده های پرتی که در جدول Describe دیده میشد میتوان به کمک متد IQR به میزان بازه ۵ صدم از دو طرف را در نظر گرفته و این نوع داده ها را در هر ستون حذف کنیم که همانطور که در شکل زیر مشاهده میشود، داده ها از ۸۹۵۰ به ۸۶۰۶ کاهش پیدا میکند که به دلیل ماهیت حذف داده های پرت و نتایج بهتر، ۳۰۰ داده حذف شده مشکلی ایجاد نمیکند:

```
columns= df.columns
for i in columns:
    Q1 = df[i].quantile(0.05)
    Q3 = df[i].quantile(0.95)
    IQR = Q3 - Q1    #IQR is interquartile range.
    filter = (df[i] >= Q1-1.5*IQR) & (df[i] <= Q3+1.5*IQR)
    df= df.loc[filter]
    print(df.shape)

```

```
(8941, 14)
(8941, 14)
(8872, 14)
(8834, 14)
(8791, 14)
(8791, 14)
(8791, 14)
(8790, 14)
(8758, 14)
(8757, 14)
(8718, 14)
(8606, 14)
(8606, 14)
(8606, 14)
```

.

نرمال سازی و کاهش ابعاد :

حال زمان آن است که داده های پردازش شده نرمال سازی شوند و به بازه میان منفی یک و یک تصویر شوند:

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
df = scaler.fit_transform(df)

df = pd.DataFrame(df, columns=['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
        'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
        'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
        'CASH_ADVANCE_FREQUENCY',
        'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
        'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE'])
df.head()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUE
0	-0.743065	-0.236364	-0.585165	-0.429431	-0.531951	-0.794362	-0.67
1	0.917756	0.144411	-0.661939	-0.597324	3.370997	-1.211598	-0.67
2	0.546192	0.525186	-0.039729	-0.597324	-0.531951	1.291814	2.74
3	0.110982	-0.997914	0.544383	-0.597324	-0.407291	-1.002981	-0.38
4	-0.334985	0.525186	-0.649063	-0.597324	-0.531951	-1.002981	-0.38

در نهایت برای انجام خوشه بندی و نتایج بهتر داده ها را به کمک روش استخراج ویژگی PCA به ۲ بعد کاهش میدهیم. این روش به کمک بیشترین میزان پراکندگی داده ها، تمامی داده هارا در جهتی که بیشترین پراکندگی هست به یک صفحه تصویر میکند (میتوان این مقدار را هر عددی حتی اعشار در نظر گرفت اما در اینجا به دلیل اینکه در ادامه به تصویر سازی داده خواهیم پرداخت، خیلی راحت تر آن است که داده ها به ۲ بعد تقلیل داده شوند نه ۳ بعد و ...)

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2, random_state=24)
df = pca.fit_transform(df)
df
```

```
array([[-1.65457978, -1.29323079],
       [-1.13172198,  3.22618307],
       [ 1.47777331, -0.08785987],
       ...,
       [-1.24254893, -1.92745102],
       [-2.45706067, -1.29888281],
       [-0.31725996, -1.03689139]])
```

```
df.shape
```

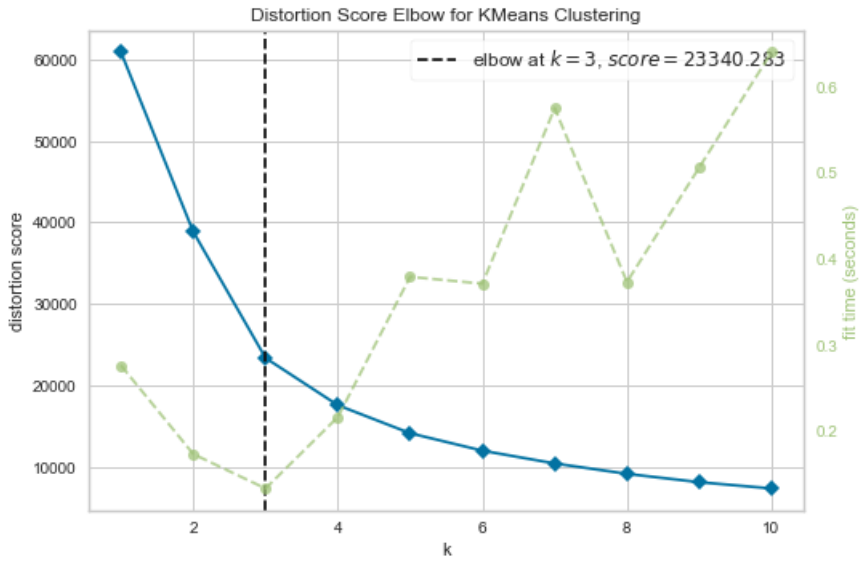
```
(8606, 2)
```

فصل سوم

پیاده سازی و مقایسه الگوریتم ها

الگوریتم K-means و معرفی مقیاس های سنجش:

ابتدا به سراغ ساده ترین و اولین الگوریتم خوشه بندی یعنی K-means میرویم . به این منظور قبل از پیاده سازی داده و خوشه بندی توسط این الگوریتم نیاز است که هایپر پارامتر آن یعنی تعداد خوشه را مشخص کنیم. به این منظور ابتدا به کمک elbow method عددی پیدا میکنیم که به منظور اطمینان این عدد را با مقیاس silhouette-score مقایسه میکنیم.



عدد تشخیص داده شده در روش زانو، ۳ میباشد.

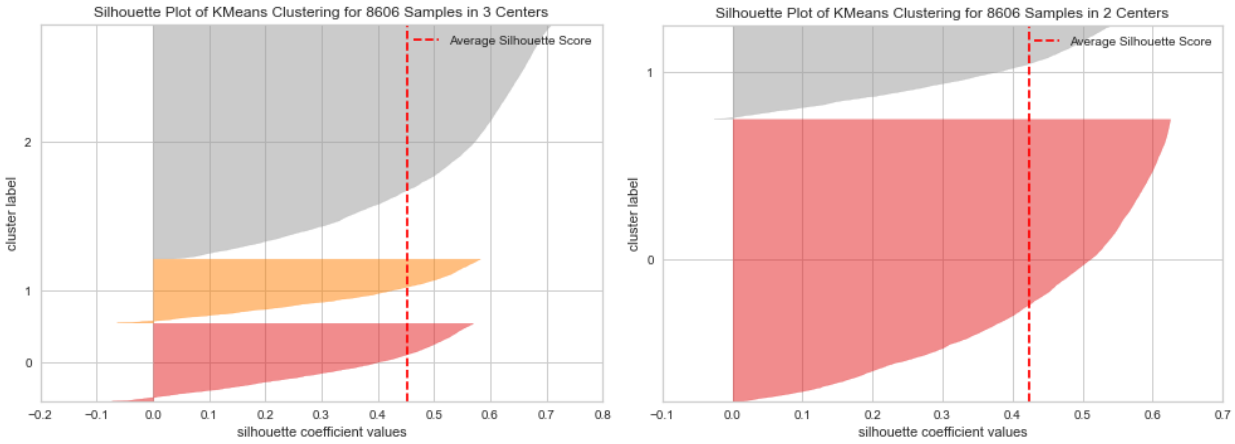
لذا در حوالی این عدد به بررسی میزان silhouette score میپردازیم:

```
Silhouette Score: 0.423
Silhouette Score: 0.453
Silhouette Score: 0.384
Silhouette Score: 0.363
```

این مقادیر به تعداد خوشه ۲ تا ۵ اشاره دارد که همانطور که مشاهده میکنید برای تعداد خوشه ۳ مطابق با روش زانو، پیشنهاد میشود. لازم به ذکر است هرچه میزان silhouette score (که مقیاسی از -۱ تا +۱

میباشد (عدد ۰ یعنی خوشه ها در هم تنیده میباشند و گسستنی نیستند و عدد منفی ۱ یعنی خوشه بندی اشتباه انجام شده است)) از عدد صفر بیشتر باشد و نزدیکتر به ۱ باشد بهتر است.

در ادامه با مصور سازی میزان این مقیاس نشان میدهیم که تعداد خوشه ۳ به تفکیک بهتری پرداخته است و میزان این مقدار برای خوشه بندی ۳ تایی بیشتر است:



پس با تعداد خوشه ۳ و به متد Kmeans++ و با تکرار معقول (n_init) به خوشه بندی کردن داده میپردازیم:

```
kmeans = KMeans(init= "k-means++", n_clusters= 3 , n_init=15 , random_state=11)

kmeans.fit(df)
print(silhouette_score(df, kmeans.labels_))
```

```
0.452940136502758
```

```
lables_kmeans= kmeans.labels_
lables_kmeans
```

```
array([1, 0, 2, ..., 1, 1, 1])
```

```
from sklearn.metrics import davies_bouldin_score, silhouette_score, calinski_harabasz_score
y_kmeans = kmeans.fit_predict(df)

print("davies_bouldin_score is: ")
print(round(davies_bouldin_score(df, y_kmeans), 4))
print("silhouette_score is: ")
print(round(silhouette_score(df, y_kmeans), 4))
print("calinski_harabasz_score is: ")
print(round(calinski_harabasz_score(df, y_kmeans), 4))
```

```
davies_bouldin_score is:
0.7808
silhouette_score is:
0.4529
calinski_harabasz_score is:
6917.8162
```

در میانه متن به توضیح مقیاس silhouette score پرداختیم اما در اینجا ۲ مقیاس دیگر نیز داریم که در زیر به توضیح مختصر هردو میپردازیم:

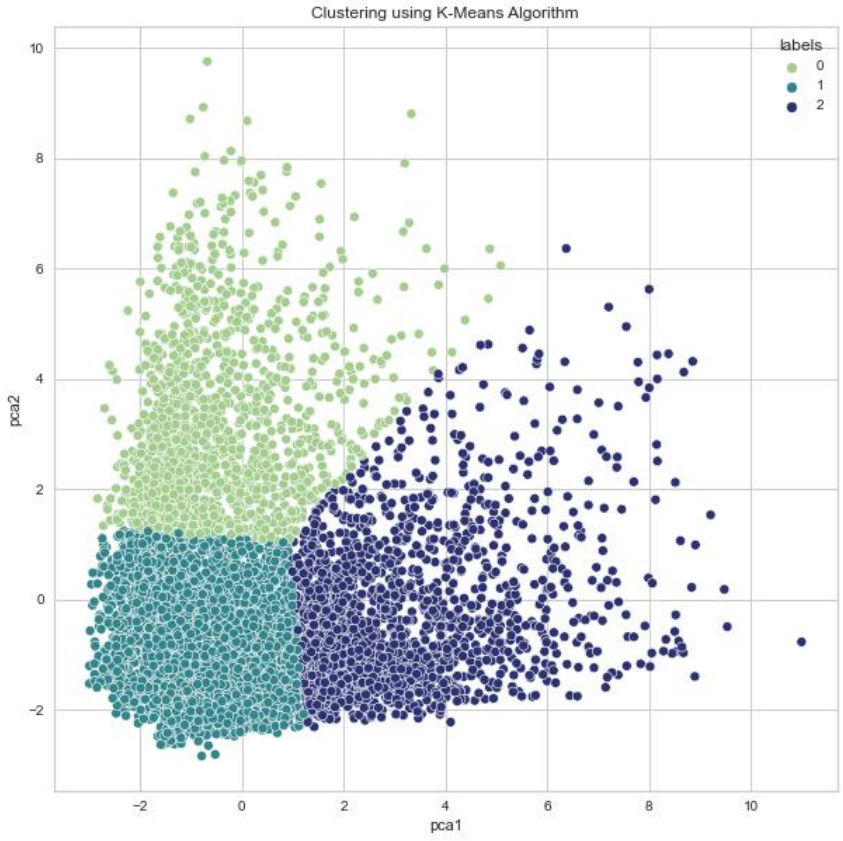
شاخص دیویس بولدین (Davies Bouldin score):

به عنوان میانگین سنجی شباهت هر خوشه با مشابه ترین خوشه تعریف می شود، که در آن شباهت نسبت فاصله های درون خوشه ای به فواصل بین خوشه ای است. بنابراین، خوشه هایی که دورتر از هم هستند و کمتر پراکنده هستند، امتیاز بهتری به دست خواهند آورد. در این مقیاس که امتیازی از صفر به بالا میدهد هرچه امتیاز به صفر نزدیک تر باشد بهتر است.

شاخص کالینسکی هاربز (Calinski harabasz):

به معیار نسبت واریانس شناخته می شود، نسبت مجموع پراکندگی بین خوشه‌ها و پراکندگی بین خوشه‌ای برای همه خوشه‌ها است، هر چه امتیاز بالاتر باشد، عملکرد بهتری خواهد داشت.

و نهایتا در ادامه کار به مصور سازی خوشه بندی میپردازیم:



الگوریتم DBscan :

این خوشه بندی مبتنی بر چگالی، نمونه‌هایی با چگالی بالا را پیدا می‌کند و خوشه‌ها را از آنها گسترش می‌دهد. برای داده هایی که دارای خوشه هایی با چگالی مشابه هستند خوب است. هایپر پارامتر های مورد بررسی در این قسمت eps و min_samples هستند. (دیفالت اپسیلون ۰,۵ است و دیفالت دیگری ۵)

به منظور تشخیص بهترین مقدار به کمک ۲ حلقه، امتیاز silhouette را برای مقادیر ۰,۲ تا ۱,۷ برای اپسیلون و مقادیر ۶ تا ۱۲ برای min_samples اجرا میکنیم:

```
for eps= 0.4
and for min_smp= 6
Silhouette Coefficient: 0.486
for eps= 0.4
and for min_smp= 7
Silhouette Coefficient: 0.289
for eps= 0.4
and for min_smp= 8
Silhouette Coefficient: 0.356
for eps= 0.4
and for min_smp= 9
Silhouette Coefficient: 0.343
for eps= 0.4
and for min_smp= 10
Silhouette Coefficient: 0.433
for eps= 0.4
and for min_smp= 11
Silhouette Coefficient: 0.320
```

```
for eps= 0.3
and for min_smp= 6
Silhouette Coefficient: 0.285
for eps= 0.3
and for min_smp= 7
Silhouette Coefficient: 0.292
for eps= 0.3
and for min_smp= 8
Silhouette Coefficient: 0.288
for eps= 0.3
and for min_smp= 9
Silhouette Coefficient: 0.338
for eps= 0.3
and for min_smp= 10
Silhouette Coefficient: 0.396
for eps= 0.3
and for min_smp= 11
Silhouette Coefficient: 0.211
```

```
for eps= 0.2
and for min_smp= 6
Silhouette Coefficient: 0.139
for eps= 0.2
and for min_smp= 7
Silhouette Coefficient: 0.093
for eps= 0.2
and for min_smp= 8
Silhouette Coefficient: 0.087
for eps= 0.2
and for min_smp= 9
Silhouette Coefficient: 0.088
for eps= 0.2
and for min_smp= 10
Silhouette Coefficient: -0.012
for eps= 0.2
and for min_smp= 11
Silhouette Coefficient: -0.020
```

```
for eps= 1.4
and for min_smp= 6
Silhouette Coefficient: 0.678
for eps= 1.4
and for min_smp= 7
Silhouette Coefficient: 0.678
for eps= 1.4
and for min_smp= 8
Silhouette Coefficient: 0.678
for eps= 1.4
and for min_smp= 9
Silhouette Coefficient: 0.663
for eps= 1.4
and for min_smp= 10
Silhouette Coefficient: 0.663
for eps= 1.4
and for min_smp= 11
Silhouette Coefficient: 0.663
```

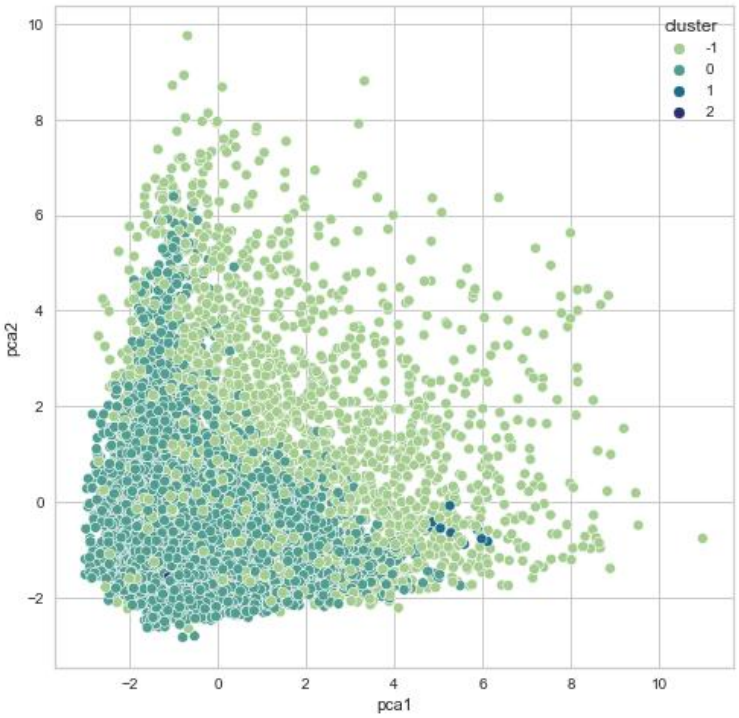
```
for eps= 0.8
and for min_smp= 6
Silhouette Coefficient: 0.621
for eps= 0.8
and for min_smp= 7
Silhouette Coefficient: 0.612
for eps= 0.8
and for min_smp= 8
Silhouette Coefficient: 0.608
for eps= 0.8
and for min_smp= 9
Silhouette Coefficient: 0.609
for eps= 0.8
and for min_smp= 10
Silhouette Coefficient: 0.622
for eps= 0.8
and for min_smp= 11
Silhouette Coefficient: 0.616
```

```
for eps= 0.6
and for min_smp= 6
Silhouette Coefficient: 0.538
for eps= 0.6
and for min_smp= 7
Silhouette Coefficient: 0.554
for eps= 0.6
and for min_smp= 8
Silhouette Coefficient: 0.588
for eps= 0.6
and for min_smp= 9
Silhouette Coefficient: 0.600
for eps= 0.6
and for min_smp= 10
Silhouette Coefficient: 0.561
for eps= 0.6
and for min_smp= 11
Silhouette Coefficient: 0.559
```

همانطور که مشاهده میشود(تمام نتایج آورده نشده است)، برای اپسیلون ۱,۴ و مینیمم سمپل ۸ این میزان به بالاترین حد خود میرسد. پس با این دو مقدار الگوریتم را اجرا کرده و نتایج را مصور سازی میکنیم :

```
from sklearn.metrics import davies_bouldin_score, silhouette_score, calinski_harabasz_score
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps = 1.4 , min_samples = 8)
y_dbscan = dbscan.fit_predict(df)

print("davies_bouldin_score is: ")
print(round(davies_bouldin_score(df, y_dbscan), 3))
print("silhouette_score is: ")
print(round(silhouette_score(df, y_dbscan), 3))
print("calinski_harabasz_score is: ")
print(round(calinski_harabasz_score(df, y_dbscan), 3))

```

```
davies_bouldin_score is:
1.023
silhouette_score is:
0.678
calinski_harabasz_score is:
19.109
```



همانطور که مشاهده میشود در هم تنیدگی خوشه ها از الگوریتم k-means بیشتر است و از نظر شخصی به نظر می آید که خوشه بندی بهتری از k-means ارائه نشده است.

الگوریتم Agglomerative Clustering :

خوشه بندی انبوهی نوعی الگوریتم خوشه بندی سلسله مراتبی(hierarchical clustering) است. یک تکنیک یادگیری ماشینی بدون نظارت است که جمعیت را به چندین خوشه تقسیم می کند، به طوری که نقاط داده در یک خوشه مشابه بیشتر و نقاط داده در خوشه های مختلف متفاوت هستند.

در این الگوریتم هایپر پارامتر های مورد بررسی linkage و تعداد خوشه است. در قسمت linkage به بررسی ۳ مورد ward، average و complete میپردازیم که در آن ward به حالتی گفتی میشود که واریانس خوشه های در حال ادغام را به حداقل می رساند، average به حالتی گفته میشود که از میانگین فواصل هر مشاهده از دو مجموعه استفاده می کند و complete به حالتی گفته میشود که از حداکثر فاصله بین تمام مشاهدات دو مجموعه استفاده می کند.

مانند الگوریتم قبلی به کمک دو حلقه به بررسی امتیاز silhouette در بازه ای از تعداد خوشه ها و ۳ حالت خوشه بندی میپردازیم :

```python
from sklearn.cluster import AgglomerativeClustering

clusters=[2,3,4,5,6,7]
linkage=['ward', 'complete', 'average']

for i in linkage:
    for j in clusters:
        agglo_model = AgglomerativeClustering(linkage=i , n_clusters=j).fit(df)
        #y_pred_agg= agglo_model.predict(df_after_norm)
        print("for linkage: ",i,"cluster: ",
              j," silhouette score: ",silhouette_score(df ,agglo_model.labels_))
```

```
for linkage:  ward cluster:  2  silhouette score:  0.3771001518492532
for linkage:  ward cluster:  3  silhouette score:  0.4041230071107595
for linkage:  ward cluster:  4  silhouette score:  0.3899414270896841
for linkage:  ward cluster:  5  silhouette score:  0.28951195820243875
for linkage:  ward cluster:  6  silhouette score:  0.29033325322739373
for linkage:  ward cluster:  7  silhouette score:  0.2963795598612192
for linkage:  complete cluster:  2  silhouette score:  0.5115701975561331
for linkage:  complete cluster:  3  silhouette score:  0.4720325940563605
for linkage:  complete cluster:  4  silhouette score:  0.32388937968589
for linkage:  complete cluster:  5  silhouette score:  0.31500173934542647
for linkage:  complete cluster:  6  silhouette score:  0.3358191077933248
for linkage:  complete cluster:  7  silhouette score:  0.32720709104186085
for linkage:  average cluster:  2  silhouette score:  0.5374387446119075
for linkage:  average cluster:  3  silhouette score:  0.5315413858224314
for linkage:  average cluster:  4  silhouette score:  0.42985141943229915
for linkage:  average cluster:  5  silhouette score:  0.4193356764505774
for linkage:  average cluster:  6  silhouette score:  0.39263906268094906
for linkage:  average cluster:  7  silhouette score:  0.3745538652576032
```

همانطور که مشاهده میشود بهترین امتیاز ها مربوط به حالت average و تعداد خوشه بندی ۲ و ۳ میباشد

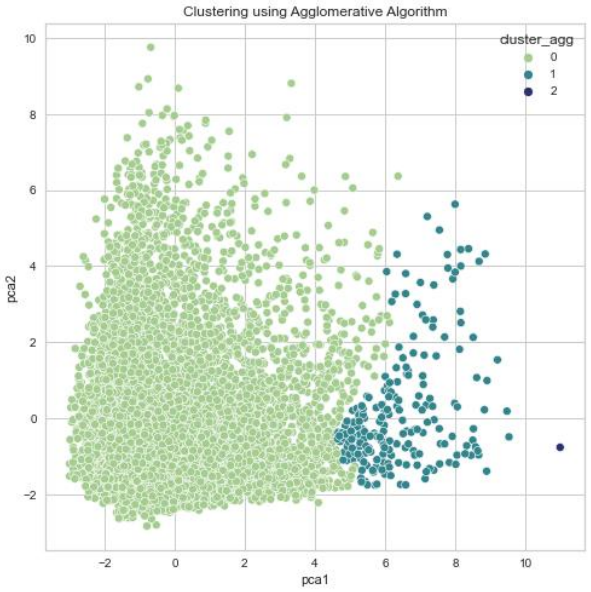
لذا برای هر دو مورد خوشه بندی انجام داده و نتایج امتیاز هارا بررسی میکنیم :

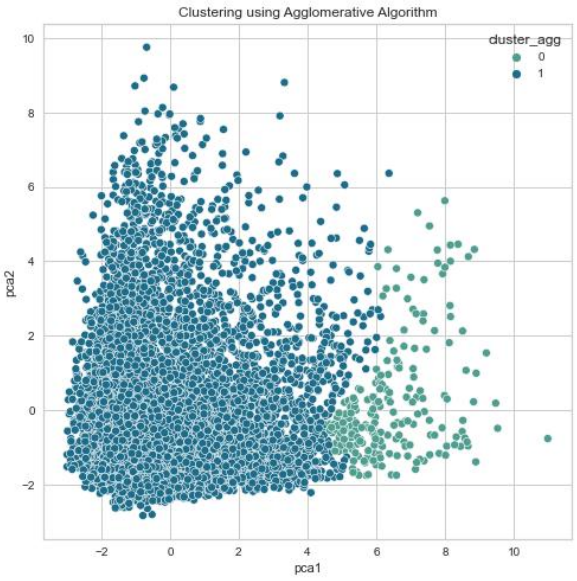
```
y_agg = AgglomerativeClustering(linkage="average" , n_clusters=2).fit_predict(df)

print("davies_bouldin_score is: ")
print(round(davies_bouldin_score(df, y_agg), 4))
print("silhouette_score is: ")
print(round(silhouette_score(df, y_agg), 4))
print("calinski_harabasz_score is: ")
print(round(calinski_harabasz_score(df, y_agg), 4))
```

```
davies_bouldin_score is:
0.5868
silhouette_score is:
0.5374
calinski_harabasz_score is:
1561.5489
```

```
y_agg = AgglomerativeClustering(linkage="average" , n_clusters=3).fit_predict(df)

print("davies_bouldin_score is: ")
print(round(davies_bouldin_score(df, y_agg), 4))
print("silhouette_score is: ")
print(round(silhouette_score(df, y_agg), 4))
print("calinski_harabasz_score is: ")
print(round(calinski_harabasz_score(df, y_agg), 4))
```

```
davies_bouldin_score is:
0.5099
silhouette_score is:
0.5315
calinski_harabasz_score is:
782.8853
```





با توجه به اینکه حالت ۳ خوشه ای امتیاز پایین تری داشت، در حالت تصویر سازی هم میبینیم که خوشه سوم فقط به چند داده پرت اختصاص پیدا کرده است و خوشه بندی ۲ تایی نتیجه بهتری دارد.

نتیجه گیری

طبق مشاهدات از ۳ الگوریتم که در طی گزارش دیدیم و جدول زیر به این نتیجه میرسیم که الگوریتم k-means به طور کلی دارای عملکرد بهتری از ۳ تای دیگر میباشد. زیرا در این الگوریتم به غیر از DBscan میزان بالایی از امتیاز silhouette را داریم و در مقایسه مقیاس دیویس بولدین نیز میزان متوسطی دارد اما در مقیاس کالینسکی هاربز بهترین میزان تفکیک خوشه ها و بالاترین میزان امتیاز را با اختلاف الگوریتم k-means دارد.

Models	Davies-Bouldin	Silhouette Score	Calinski-Harabasz
K-Means	0.780800	0.452900	6917.816200
DBSCAN	1.023000	0.678100	19.109000
Agglomerative-3	0.509900	0.531500	782.885300
Agglomerative-2	0.586800	0.537400	1561.548900

منابع و مراجع:

- https://www.kaggle.com/code/aninditapani/clustering-with-pca
- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN

پایان